Alexander Garcia June 12, 2017

Assignment-3

1. (a) **False**.

Because $\mathbf{A}$ is nonsingular, there are a number of properties that are associated with it. One of these is the fact that $\mathbf{A}$ being singular implies that $\mathbf{Ax} = \mathbf{b}$ has a unique solution for all $\mathbf{b}$. Likewise if $\mathbf{b} = \mathbf{0}$, then $\mathbf{x}$ is necessarily the 0 vector.

This question was done assuming that $\mathbf{A} = n \times n \rightarrow \mathbf{b} = 1 \times n$. If $\mathbf{b}$ is of size $m < n$, then $\mathbf{Ax} = \mathbf{b}$ has an infinity of solutions. However, this does not depend on the contents of $\mathbf{b}$, but rather the initial parameters of the problem.

(b) **False**.

This can be proven false simply by taking the determinant of a 3×3 matrix.

$$det(\mathbf{A}) = a_{11}(a_{22}a_{33} + a_{23}a_{32}) - a_{12}(a_{21}a_{33} + a_{23}a_{31}) + a_{13}(a_{21}a_{32} + a_{22}a_{31})$$

As there are terms in this equation that are not dependant on a_{ii}, a matrix with a zero on the principle diagonal is not necessarily singular.

(c) **False**.

The conditioning of a matrix $\mathbf{A}$ is related to the largest sum of a single column (the 1-norm). The partial pivoting algorithm scans the rest of the column below the current pivot point (a_{kk}) for the largest value.

A well-conditioned matrix would simply have small column sums, which does not necessarily affect the location of the largest values in each row, which in turn does not affect the need for partial pivoting.

(d) **True**.

By definition, the condition number of a matrix $\mathbf{A} = ||\mathbf{A}||\ ||\mathbf{A^{-1}}||$. Therefore, the condition number of $\mathbf{A^{-1}} = ||\mathbf{A^{-1}}||\ ||(\mathbf{A^{-1}})^{-1}|| = ||\mathbf{A^{-1}}||\ ||\mathbf{A}||$. As $||A||$, $||A^{-1}||$ are scalar values, the two products are the same.

(e) **False**.

When working with a linear system, the solution to the system is the point at which every line defined by matrix $\mathbf{A}$ intersects. If there is no point where every line intersects, then there are no solutions. Because the system is linear, two lines can have only one discreet intersection. Therefore, it is not possible for 2 (or more) lines to intersect at more than one point, so there must be either exactly 1 solution, or infinitely many solutions.

(f) One of the reasons pivoting is essential in the G.E. method of solving matrices is to avoid potential zeroes on the principle diagonal. When the G.E. is being done, each element on the principle diagonal determines the multiplier for the rows below. $m_{ij} = \frac{a_{ij}}{a_{jj}}$. Since

the diagonal element is in the denominator, it must be nonzero. This is a mathematical reason for pivoting.

The other reason is a numerical one. If the current pivot point is a_{jj}, and this value is very small (10^{-16}), then the multiplier that results will be extremely large. If the values in column j are then relatively small to a_{jj}, the subtraction will result in a loss of significance, due to not being able to express numbers such as $10^{16} - 1$ accurately. Partial pivoting moves the row with the largest j^{th} element to the current row, then computes the multiplier. This avoids the large multiplier result, and will aid in maintaining significance.

(g) The condition number is a nontrivial computation due to the computation of the norms that make up the number. In order compute the 1-norm, which is common, the algorithm must sum every column, the largest of which becomes the condition number. This alone requires $O(n^2)$ operations. The algorithm must then compute the inverse of $\mathbf{A}$, which is an even more expensive operation. More curcially, the inverse of matrix $\mathbf{A}$ must be completely recomputed if $\mathbf{A}$ is even slightly modified.

(h) On its own, the relative residual does not carry much weight. A small residual would, at firs glance, seem to imply low error, and therefore higher accuracy. However, this is not the case, and it turns out that the condition number of a matrix is directly involved in the accuracy of the solution. In order for a solution to be accurate, the solution should not only have a small relative residual, but should also be moderately well conditioned.

(i) i. $det(\mathbf{cA}) = c^n * det(\mathbf{A})$
In this case, $\mathbf{A}$ would be an $n \times n$ diagonal matrix, with every diagonal entry = 1, and $c = \frac{1}{2}$. The determinant of a diagonal matrix with every diagonal value = 1 is simply the identity matrix , and $det(\mathbf{I})$ is 1. Therefore, the determinant of the matrix $\frac{1}{2}\mathbf{I} = \frac{1}{2}^n$.

ii. Assume the use of the 1-norm
$||\mathbf{A}||_1 = \frac{1}{2}$
Because the inverse of a diagonal matrix is found by simply inverting all diagonal elements, $||\mathbf{A}^{-1}||_1 = 2$
$K(\mathbf{A}) = ||\mathbf{A}||_1 * ||\mathbf{A^{-1}}||_1 = 1$

iii. Any scalar multiple of the identity matrix will not amplify input errors. This makes intuitive sense, as the condition number of a matrix sets an upper bound on the maximum distortion of the result due to input errors. Because each element in the system $(x_1, x_2, \ldots, x_n)$ only appears once in $\mathbf{A}$, each entry simply takes the corresponding $\mathbf{b}$ value $(a_1 = b_1, a_2 = b_2, \ldots, a_n = b_n)$. Any error that is present in the input will be directly translate to the output.

2. (a)

$$\begin{bmatrix} 1 & 2 \\ 2 & 4.01 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 3 \\ 6.01 \end{bmatrix}$$

$x_1 = 3 - 2x_2$
$2(3 - 2x_2) + 4.01x_2 = 6.01$
$6 + 0.01x_2 = 6.01$
$0.01x_2 = 0.01$
$x_2 = 1$
$x_1 = 1$

(b)

$$\begin{bmatrix} 1 & 2 \\ 2 & 4.01 \end{bmatrix} \begin{bmatrix} -600 \\ 301 \end{bmatrix} = \begin{bmatrix} 2 \\ 7.1 \end{bmatrix}$$

Here,

$$x^* = \begin{bmatrix} -600 \\ 301 \end{bmatrix}$$

and

$$b^* = \begin{bmatrix} 2 \\ 7.1 \end{bmatrix}$$

The relative forward error is defined by $\frac{||\mathbf{x}^*||_\infty}{||\mathbf{x}||_\infty}$, while the relative backward error is $\frac{||\mathbf{b}^*||_\infty}{||\mathbf{b}||_\infty}$.

Relative Forward Error	Relative Backward Error
$\|\|\mathbf{x}^*\|\| = 600$	$\|\|\mathbf{b}^*\|\| = 7.1$
$\|\|\mathbf{x}\|\| = 1$	$\|\|\mathbf{b}\|\| = 6.01$
$\frac{600}{1} = 600$	$\frac{7.1}{6.01} \approx 1.18\ldots$

The error magnification factor is (relative forwards error $\div$ relative backwards error).
$E_m = 600 \div \frac{7.1}{6.01} \approx 507.89\ldots$

(c) $K(\mathbf{A}) = ||\mathbf{A}||\ ||\mathbf{A^{-1}}||$
$||\mathbf{A}|| = 6.01$

$$\mathbf{A^{-1}} = \frac{1}{0.01} \begin{bmatrix} 4.01 & -2 \\ -2 & 1 \end{bmatrix}$$

$||\mathbf{A^{-1}}|| = 201$

$$K(\mathbf{A}) = 201 * 6.01 = 1208.01$$

The poor conditioning of the number is a clear indication that a small change to the data will cause a large change in the result. In this case, the altered data that gave rise to the vector $\mathbf{x}^*$ was $\mathbf{b}^*$. The large condition number sets the upper bound on the relative output error, meaning that $\mathbf{x}^*$ can grow extremely large, without producing extremely different results in $\mathbf{b}^*$.

3. (a) Begin with the augementd matrix $\mathbf{A}|\mathbf{b}$

$$\mathbf{A}|\mathbf{b} = \begin{bmatrix} 1 & 2 & 3 & 4 & -6 \\ 2 & 3 & 7 & 9 & -17 \\ -2 & -6 & -2 & -4 & -4 \\ 1 & 0 & 11 & 15 & -34 \end{bmatrix}$$

Step 1: Pivot point a_{11}
First, we must employ partial pivoting to ensure there are no multipliers > 1. We see that row 2 has the largest value in column 1, so we use this as the first fow instead. The permutation matrix associated with swapping rows $R1$ and $R2$ is

$$\mathbf{P} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and the matrix we are now operating on is

$$\mathbf{A} \begin{bmatrix} 2 & 3 & 7 & 9 & -17 \\ 1 & 2 & 3 & 4 & -6 \\ -2 & -6 & -2 & -4 & -4 \\ 1 & 0 & 11 & 15 & -34 \end{bmatrix}$$

We now calculate the multipliers needed to zero out the first column.

$$m_{21} = \frac{a_{21}}{a_{11}} = 0.5 \;\Big\|\; m_{31} = \frac{a_{31}}{a_{11}} = -1 \;\Big\|\; m_{41} = \frac{a_{41}}{a_{11}} = 0.5$$

Then, carry out the row operations on $R2, R3, R4$.

$$R2 \to R2 - m_{21}R1 \;\Big\|\; R3 \to R3 - m_{31}R1 \;\Big\|\; R4 \to R4 - m_{41}R1$$

The new matrix generatred after these operations is

$$\mathbf{A} = \begin{bmatrix} 2 & 3 & 7 & 9 & -17 \\ (0.5) & 0.5 & -0.5 & -0.5 & -2.5 \\ (-1) & -3 & -5 & 5 & -21 \\ (0.5) & -1.5 & 7.5 & 10.5 & -25.5 \end{bmatrix}$$

where the normal zeros are replaced by the multipliers.

Step 2: Pivot point a_{22}
Again, start by swapping rows according to partial pivoting. This includes changing the positions of the multipliers, although their values will not be affected by the row operations that follow. It also includes swapping the corresonding rows in the permutation matrix. In this case, we swap rows $R2$ and $R3$

$$\mathbf{A} = \begin{bmatrix} 2 & 3 & 7 & 9 & -17 \\ (-1) & -3 & -5 & 5 & -21 \\ (0.5) & 0.5 & -0.5 & -0.5 & -2.5 \\ (0.5) & -1.5 & 7.5 & 10.5 & -25.5 \end{bmatrix}$$

and

$$\mathbf{P} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Next, calculate the new multipliers. In this step there are only two, as the only rows that need have items zeroed out are $R3$ and $R4$

$$m_{32} = \frac{a_{32}}{a_{22}} = -\frac{1}{6} \;\Big\|\; m_{42} = \frac{a_{42}}{a_{22}} = 0.5$$

Next, perform the row operations on $R3, R4$

$$R3 \to R3 - m_{32}R2 \;\Big\|\; R4 \to R4 - m_{42}R2$$

The new matrix is

$$\mathbf{A} = \begin{bmatrix} 2 & 3 & 7 & 9 & -17 \\ (-1) & -3 & -5 & 5 & -21 \\ (0.5) & (-\frac{1}{6}) & \frac{1}{3} & \frac{1}{3} & -6 \\ (0.5) & (0.5) & 5 & 8 & 36 \end{bmatrix}$$

again, with the normal zeros replaced by the multipliers.

Step 3: Pivot point a_{33}
In this case, we have to switch $R3$ and $R4$ in $\mathbf{A}$ and $\mathbf{P}$ according to partial pivoting.

$$\mathbf{A} = \begin{bmatrix} 2 & 3 & 7 & 9 & -17 \\ (-1) & -3 & -5 & 5 & -21 \\ (0.5) & (0.5) & 5 & 8 & 36 \\ (0.5) & (-\frac{1}{6}) & \frac{1}{3} & \frac{1}{3} & -6 \end{bmatrix}$$

and

$$\mathbf{P} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

The only multiplier to be calculated is m_{43}

$$m_{43} = \frac{a_{43}}{a_{33}} = \frac{1}{15}$$

and the only row operation to be performed is

$$R4 \to R4 - m_{43}R3$$

The final matrix is

$$\mathbf{A} = \begin{bmatrix} 2 & 3 & 7 & 9 & -17 \\ (-1) & -3 & -5 & 5 & -21 \\ (0.5) & (0.5) & 5 & 8 & 36 \\ (0.5) & (-\frac{1}{6}) & (\frac{1}{15}) & -0.2 & -8.4 \end{bmatrix}$$

From here, we can extract both the lower and upper triangular matrices $\mathbf{L}, \mathbf{U}$ respectively.

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0.5 & 0.5 & 1 & 0 \\ 0.5 & -\frac{1}{6} & \frac{1}{15} & 1 \end{bmatrix} \mathbf{U} = \begin{bmatrix} 2 & 3 & 7 & 9 \\ 0 & -3 & 5 & 5 \\ 0 & 0 & 5 & 8 \\ 0 & 0 & 0 & -0.2 \end{bmatrix} \mathbf{P} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

(b) When checked in MATLAB, the results of the hand calculation agree with the computed result.

`lup_check.m` script used to generate matrices.

```
% generate LU decomposition of matrix A

A = [1 2 3 4; 2 3 7 9; -2 -6 -2 -4; 1 0 11 15];
[L,U,P] = lu(A);


%Display results
fprintf('L =\n');
disp(L);

fprintf('\nU =\n');
disp(U);

fprintf('\nP =\n');
disp(P);
```

Output of `lup_check.m`

```
lup_check
L =
    1.0000         0         0         0
   -1.0000    1.0000         0         0
    0.5000    0.5000    1.0000         0
    0.5000   -0.1667    0.0667    1.0000

U =
    2.0000    3.0000    7.0000    9.0000
         0   -3.0000    5.0000    5.0000
         0         0    5.0000    8.0000
         0         0         0   -0.2000

P =
     0     1     0     0
     0     0     1     0
     0     0     0     1
     1     0     0     0
```

(c) If the decomposition is correct, $\mathbf{LU} == \mathbf{AP}$

$$\mathbf{LU} = \begin{bmatrix} L_{11}U_{11} & L_{11}U_{12} & L_{11}U_{13} & L_{11}U_{14} \\ L_{21}U_{11} & L_{21}U_{12} + L_{22}U_{22} & L_{21}U_{13} + L_{22}U_{23} & L_{21}U_{14} + L_{22}U_{24} \\ L_{31}U_{11} & L_{31}U_{12} + L_{32}U_{22} & L_{31}U_{13} + L_{32}U_{23} + L_{33}U_{33} & L_{31}U_{14} + L_{32}U_{24} + L_{33}U_{34} \\ L_{41}U_{11} & L_{41}U_{12} + L_{42}U_{22} & L_{41}U_{13} + L_{42}U_{23} + L_{43}U_{33} & L_{41}U_{14} + L_{42}U_{24} + L_{43}U_{34} + L_{44}U_{44} \end{bmatrix}$$

$$= \begin{bmatrix} 2 & 3 & 7 & 9 \\ -2 & (-3-3) & (-7+5) & (-9+5) \\ 1 & (1.5-1.5) & (3.5+2.5+5) & (4.5+2.5+8) \\ 1 & (1.5+0.5) & (3.5-\frac{5}{6}+\frac{1}{3}) & (4.5-\frac{5}{6}+\frac{8}{15}-0.2) \end{bmatrix}$$

$$= \begin{bmatrix} 2 & 3 & 7 & 9 \\ -2 & -6 & -2 & -4 \\ 1 & 0 & 11 & 15 \\ 1 & 2 & 3 & 4 \end{bmatrix}$$

The product $\mathbf{AP}$ can be found analytically by taking the row operations needed to convert the identity matrix $\mathbf{I}$ into $\mathbf{P}$, and applying them to $\mathbf{A}$.
$\mathbf{P} = (R1_I \leftrightarrow R2_I) \rightarrow (R2_I \leftrightarrow R3_I) \rightarrow (R3_I \leftrightarrow R4_I)$
Performing the same operations on $\mathbf{A}$ yields

$$\mathbf{AP} = \begin{bmatrix} 2 & 3 & 7 & 9 \\ -2 & -6 & -2 & -4 \\ 1 & 0 & 11 & 15 \\ 1 & 2 & 3 & 4 \end{bmatrix}$$

The two matrices are equal, and there is no residual.

4. The matrix $\mathbf{A}$ is symmetric positive definite if the determinants of all leading principle submatrices are positive.

$$\mathbf{A} = \begin{bmatrix} 4 & -2 & 0 \\ -2 & 2 & -3 \\ 0 & -3 & 10 \end{bmatrix}$$

Here, the determinant of the first principle submatrix is simply 4, the second is $((4*2) - (-2*-2)) = 4$, and the third is $4((2*10) - (-3*-3)) + 2((-2*10) - 0) + 0 = 4$.

The Cholesky decomposition says that $\mathbf{LL^T} = \mathbf{A}$, meaning

$$\begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & l_{31} \\ 0 & l_{22} & l_{32} \\ 0 & 0 & l_{33} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

Step 1: Multiply $R1$ of $\mathbf{L}$ with columns of $\mathbf{L^T}$. This result is the first row of $\mathbf{A}$.

$(l_{11})^2 = 4 \quad l_{11} = 2$
$l_{11}l_{21} = -2 \quad l_{21} = -1$
$l_{11}l_{31} = 0 \quad l_{31} = 0$

Step 2: Multiply $R2$ of $\mathbf{L}$ with columns of $\mathbf{L^T}$. This result is the second row of $\mathbf{A}$. Since the first equation $(l_{21}l_{11})$ does not have any unknowns, we can skip this calculation.

$(l_{21})^2 + (l_{22})^2 = 2 \quad l_{22} = 1$
$l_{21}l_{31} + l_{22}l_{32} = -3 \quad l_{32} = -3$

Step 3: Multiply $R3$ of $\mathbf{L}$ with columns of $\mathbf{L^T}$. This result is the second row of $\mathbf{A}$. Since The first equation $(l_{32}l_{11})$ and the second equation $(l_{31}l_{21} + l_{32}l_{22})$ do not have any unkowns, we can skip them, and are left with a single calculation for l_{33}.

$(l_{31})^2 + (l_{32})^2 + (l_{33})^2 = 10 \quad l_{33} = 1$

$$\mathbf{L} = \begin{bmatrix} 2 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & -3 & 1 \end{bmatrix}$$

Just to make sure that this is a valid Cholesky decomposition,

$$\begin{bmatrix} 2 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & -3 & 1 \end{bmatrix} \begin{bmatrix} 2 & -1 & 0 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} (2*2) & (2*-1) & 0 \\ (2*-1) & (-1*-1)+(1*1) & (-3*1) \\ 0 & (-3*1) & (-3*-3)+(1*1) \end{bmatrix}$$

$$= \begin{bmatrix} 4 & -2 & 0 \\ -2 & 2 & -3 \\ 0 & -3 & 10 \end{bmatrix}$$

5. (a) See below

`myA.m` function

```
% myA.m
%
% Fill out an nxn matrix, where each entry a_{ij} is determined by the
%   formula a_{ij} = 5/(i+2j-1). Return the result b=Ax, where x is a size
%   n vector of ones.

function [A,b] = myA(n)
% input:
% n = size of square matrix A
%
% output:
```

```
% A = resulting matrix A
% b = Ax

% initialize A as a zero matrix
A = zeros(n);

% loops to populate A
for i = 1:n
    for j = 1:n
        A(i,j) = 5/(i+2*j-1);
    end
end

% create x vector
x = ones(n,1);

% calculate b vector
b = A*x;
```

(b) The results of `myA.m` for size $n = 5:20$ are generated by a second script, `runmyA.m`. To summarize, the condition number of $\mathbf{A}$ ($K(\mathbf{A})$) grows increasingly large with n. This causes the relative error to increase as well, causing the significance of the result to deteriorate, until there is no significance left at $n = 17$.

Note: The `fprintf` command that displays the results is too long to be displayed on a single line. The ending variables that are used are `nE, kA, rr, scaledR`, which are the norm of the error, the condition number of $\mathbf{A}$, the relative residual, and the upper bound on the error, respectively.

`runmyA.m`

```
% Script to run myA.m over values of n = 5:20, or unil the error grows
%   large enough that there is no significance in the results
clc;

%supress warnings
warning('off','all');

for n = 5:20
    % get calculated values of the solution vector x
    [A,b] = myA(n);
    xc = A\b;

    % define correct solution, x
    x = ones(n,1);

    % calculate residual
    r = b - A*xc;
    % then the infinity norm of the residual
```

```
    nr = norm(r,Inf);

    % calculate error
    E = xc - x;
    % then the infinity norm of the error
    nE = norm(E,Inf);

    % estimate condition number of A
    % (this should grow with increasing n)
    kA = cond(A,Inf);

    % calculate relative error
    rE = nE / norm(xc,Inf);

    % calculate relative residual
    rr = nr / (norm(A,Inf)*norm(xc,Inf));

    % calues values for error bound inequality
    scaledR = kA * rr;

    % display results
    fprintf('\nn = %d\n',n);
    fprintf('computed solution\n');
    disp(xc);
    fprintf('residual norm  error norm     K(A)           relative residual
upper error bound\n');
    fprintf('%10.5e    %10.5e    %10.5e    %10.5e        %10.5e     %10.5e\n',nr,nE
    fprintf('\nrelative error\n');
    fprintf('%10.5e\n',rE);

    % stop the loop if re > 1
    if rE >= 1
        fprintf('Error has exceeded 100%%.\n');
        return;
    end

end
```

Output of `runmyA.m` script

```
n = 5
computed solution
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000

residual norm  error norm     K(A)           relative residual  upper error bound
8.88178e-16    2.75980e-11    1.97907e+06    1.55593e-16        3.07930e-10
```

```
relative error
2.75980e-11

n = 6
computed solution
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000

residual norm  error norm     K(A)           relative residual  upper error bound
8.88178e-16    8.66516e-11    7.03420e+07    1.45009e-16        1.02002e-08
relative error
8.66516e-11

n = 7
computed solution
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000

residual norm  error norm     K(A)           relative residual  upper error bound
8.88178e-16    2.57584e-08    2.44925e+09    1.37019e-16        3.35594e-07
relative error
2.57584e-08

n = 8
computed solution
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000

residual norm  error norm     K(A)           relative residual  upper error bound
4.44089e-16    1.39412e-07    9.72859e+10    6.53587e-17        6.35848e-06
relative error
1.39412e-07

n = 9
computed solution
    1.0000
```

```
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000

residual norm  error norm     K(A)           relative residual  upper error bound
8.88178e-16    3.42030e-05    3.65077e+12    1.25580e-16        4.58464e-04
relative error
3.42021e-05

n = 10
computed solution
    1.0000
    1.0000
    1.0000
    1.0000
    0.9999
    1.0004
    0.9992
    1.0009
    0.9995
    1.0001

residual norm  error norm     K(A)           relative residual  upper error bound
1.77636e-15    9.10138e-04    1.31367e+14    2.42371e-16        3.18397e-02
relative error
9.09311e-04

n = 11
computed solution
    1.0000
    1.0000
    1.0000
    1.0004
    0.9964
    1.0162
    0.9570
    1.0693
    0.9337
    1.0346
    0.9924

residual norm  error norm     K(A)           relative residual  upper error bound
8.88178e-16    6.93016e-02    4.80162e+15    1.10020e-16        5.28273e-01
relative error
6.48101e-02
```

```
n = 12
computed solution
     1.0000
     1.0000
     0.9998
     1.0039
     0.9601
     1.2285
     0.2077
     2.7222
    -1.3593
     2.9761
     0.0764
     1.1844

residual norm  error norm     K(A)           relative residual  upper error bound
1.77636e-15    2.35928e+00    1.91125e+17    7.69365e-17        1.47045e+01
relative error
7.92745e-01

n = 13
computed solution
     1.0000
     1.0000
     1.0002
     0.9961
     1.0461
     0.6916
     2.2721
    -2.3804
     6.8928
    -5.6958
     5.7738
    -0.9384
     1.3419

residual norm  error norm     K(A)           relative residual  upper error bound
8.88178e-16    6.69576e+00    8.65998e+17    1.62077e-17        1.40358e+01
relative error
9.71419e-01

n = 14
computed solution
     1.0000
     1.0000
     1.0007
     0.9829
     1.1976
    -0.2557
     5.6692
    -9.1271
```

```
   12.1257
    0.8049
  -14.8577
   21.1347
  -10.0368
    3.3615

residual norm  error norm     K(A)           relative residual  upper error bound
1.77636e-15    2.01347e+01    3.41375e+18    1.03396e-17        3.52968e+01
relative error
9.52684e-01

n = 15
computed solution
    1.0000
    1.0000
    1.0005
    0.9891
    1.1084
    0.4433
    2.4592
   -0.4353
   -0.2028
    2.5544
   10.9906
  -26.7285
   31.7774
  -15.4747
    4.5184

residual norm  error norm     K(A)           relative residual  upper error bound
1.33227e-15    3.07774e+01    1.50673e+18    5.05391e-18        7.61488e+00
relative error
9.68531e-01

n = 16
computed solution
    1.0000
    1.0000
    0.9988
    1.0307
    0.6059
    3.7977
  -10.6929
   29.8327
  -36.6665
   11.3318
   38.5622
  -38.3413
  -15.5495
   54.5971
```

```
  -34.7402
    9.2333

residual norm  error norm     K(A)           relative residual  upper error bound
2.66454e-15    5.35971e+01    3.72142e+18    5.77433e-18        2.14887e+01
relative error
9.81684e-01

n = 17
computed solution
    1.0000
    1.0000
    1.0005
    0.9880
    1.1435
    0.0684
    4.4167
   -5.5352
    3.8331
   14.7443
  -25.8951
    9.6909
   31.4856
  -45.1613
   29.2959
   -6.6839
    1.6085

residual norm  error norm     K(A)           relative residual  upper error bound
1.77636e-15    4.61613e+01    2.61016e+18    4.57427e-18        1.19396e+01
relative error
1.02214e+00
Error has exceeded 100%.
```